DOCUMENT RESUME

ED 137 088 SE 022 181

AUTHOR Shymansky, James A.; And Others
TITLE A Computer Program Designed to Identify Behavior Patterns in Observational Data. Technical Report 12.
INSTITUTION Iowa Univ., Iowa City. Science Education Center.
PUB DATE Feb 77
NOTE 74p.; Not available in hard copy due to marginal legibility of original document

EDRS PRICE MF-$0.83 Plus Postage. HC Not Available from EDRS.
DESCRIPTORS *Behavior Patterns; Classroom Environment; *Classroom Observation Techniques; *Teacher Behavior; *Teacher Evaluation

ABSTRACT

The technique of macroanalysis has been developed to facilitate the process of examining patterns of behavior. In this technique, sequentially recorded observational data are computer-analyzed in units of three or more codes. Behavior patterns that have been identified from observational data are collected so that the sequence of individual behaviors (codes) is preserved. The analyst decides the pattern length, which may vary from groups of one to five or more successive codes in the data. He/she also has the option of formulating patterns which include repetitive codes or of collapsing the repetitive codes. Collapsing codes reduces strings of repetitive codes into a single code. The following kinds of information are provided in the pattern analysis: pattern identification, listing options according to frequency or beginning character in the pattern, frequency and percentage of patterns, and raw data when the collapsing option is specified. User information, summary sheet of program options, and a sample printout are included. (CS)

****
* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *
****

Technical Report Series

ED137088

U S DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

SE 022 181

ence Education Center, The University of Iowa, Iowa City, Iowa 52242

SCIENCE EDUCATION CENTER

The University of Iowa

February 1977

technical report 12

A Computer Program Designed to Identify Behavior Patterns in Observational Data

by
James A. Shymansky
John E. Penick
Jay D. Wortman

James A. Shymansky
Associate Professor

John E. Penick
Assistant Professor

Jay D. Wortman

Science Education Center
The University of Iowa
Iowa City, Iowa 52242

A Computer Program Designed to Identify Behavior Patterns in Observational Data

technical report 12

Introduction

Macroanalysis is a technique by which sequentially recorded observational data are analyzed in units of three or more individual codes (Campbell, 1973). In dealing with chains of three or more behavior codes, macroanalysis facilitated the examination of "patterns of behaviors;" patterns which are totally ignored in conventional frequency measures and only slightly accounted for in the more sophisticated matrix analyses.

Early studies of classroom climate such as those by Withall (1949, 1951) and Flanders (1965) attempted to deal with behavior patterns in the classroom; however, due to inadequate techniques for positively identifying the exact patterns of behavior, gross descriptors such as "learner-centered" vs "teacher-centered" (Withall) and "direct" vs "indirect" (Flanders) were employed. These broad definitions of behavior have contributed to the inconsistent findings across studies of seemingly similar strategies and have lead to criticism of interaction analysis as a research tool by users in the field (Rosenshine, 1970). Even with such criticism, the need to study behavior patterns in the classroom, especially in terms of establishing teacher effectiveness, is a recognized fact (Smith, 1967). With the introduction of macroanalytic techniques in more recent years, interests in behavior patterns and strategies of teaching and learning in the classroom have been rekindled. Studies by Shymansky, et al. (1975); Penick,

et al. (1976); and Campbell (1973) in which macroanalytic procedures were employed have been successful in identifying definite strategies or patterns of learning and teaching in various classroom settings.

Unfortunately, the search for behavior patterns in the classroom and elsewhere through the use of macroanalytic techniques has been limited to only a handful of researchers due to the complicated nature of the computer programming required in the analysis. The program described in the following pages is designed to identify the behavior patterns occurring in sequentially recorded observational data.

General Program Description

Behavior patterns can be identified from any observational data which are collected in a manner which preserves the sequence of the individual codes. Although the basic program function is the identification of patterns, several types of patterns as well as several characteristics of the patterns are built into the program or can be called at the user's option. Following is a list of the main functions and options contained in the program:

Pattern Length:

The behavior patterns are formed by grouping successive codes in the raw data. These groupings can be specified to contain from one to five characters (Note: a grouping size or pattern length of 1 amounts to a simple frequency report of individual codes

while a pattern length of 2 is equivalent to a conventional pairs analysis characteristic of the matrix procedures).

Pattern Type:

Once the pattern length is decided, the user has the option of formulating patterns which include repetitive characters or collapsing repetitive characters within the sequence. The latter process of collapsing is referred to as MACROanalysis (Shymansky, et al., 1975). The collapsing function has the effect of reducing long strings of repetitive codes (1) into a single code within behavior patterns. For example, without collapsing, the pattern AXXAB is possible from the data sequence AXXABAACX, etc. Incorporating the collapsing function prohibits the repetition of any code within the patterns generated. Thus, the first 5-code pattern identified in the above sequence is AXABA.

Pattern Listing:

Once all patterns within the data are identified, the user can specify the amount of data to be printed and the form of the listing. During the actual program execution, all patterns identified are stored and counted. Obviously, not all patterns will occur with the same frequency. Thus, the user may opt to have the top 100 patterns printed only. Furthermore, the user may opt to have these patterns listed according to frequency or beginning character in the pattern.

Miscellaneous Output:

In addition to the program options listed above, several other pieces of information are provided in the pattern analysis. The frequency of occurrence of all individual behavior codes is provided as well as the percentage of occurrence. For each pattern listed, the frequency and percentage of all patterns identified is provided. When the collapsing option is specified, the average length of the span of codes comprising the raw data from which the collapsed sequence was derived as well as the average number of each individual code within the collapsed span are listed for each collapsed pattern identified.

A sample printout is included in this report to clarify several of the user options and the output provided. Specifications for calling each of the options and other input data are also discussed.

User Information

Following is a list of "Keywords" used in the program to specify pattern parameters. A summary sheet of the program options is contained in Appendix I.

Input Options:

These commands specific where the data are located on the input cards.

(a) BEGIN_FIELD -- This specifies the position of the first character of data on the input card which will

initiate the first behavior pattern. For data beginning in column 21, the control card would contain the message:

BEGIN_FIELD = 21

(b) BEGIN_LENGTH -- This specifies the length of the data field on the input card. In the case of data contained in columns 21-80, the control card would contain the message:

BEGIN_LENGTH = 60

(c) FIELD_SPAN -- This command specifies which characters of the field are to be used in forming the patterns. For example, if the data were contained in columns 21-80 and only the first character of each group of 3 codes were to be used in forming the patterns, i.e., data in column 21, 24, 27, 30, etc. the control card message would be

FIELD_SPAN = 3

If every character were to be used in formulating the pattern, the message would be

FIELD_SPAN = 1

Pattern Options:

As was mentioned earlier, the user has the option of choosing pattern length and collapsing procedures.

(a) PATLEN -- To specify pattern length from 2-5, the following control message is used:

PATLEN = 5

(b) COLLAPSE -- Patterns generated from the raw observational data can be specified in one of two forms: (1) collapsed form in which repetitive codes in the raw data are contracted to a single unit code in the identification of patterns, and (2) repetitive form in which repetitive codes within the raw data are preserved and recognized in the patterns. Collapsing reduces the raw code sequecne AAABBAADB to a five code pattern of ABADB while retaining the repetitive codes results in the five code patterns AAABB, AABBA, ABBAA, BBAAD, and BAADB. To activate the collapsing option, the following control card message is used:

COLLAPSE = 'YES'

A default function in the program will specify that repetitive codes be preserved if the COLLAPSE = 'YES' command is deleted.

Input Character Options:

(a) VALID_CHAR -- This control card contains a listing of all the characters which will be used to create behavior patterns.

(b) INVALID_CHAR -- This card lists the characters which can occur in the data field but which are not to be used in formulating behavior patterns. For example, a data field may purposely contain blanks or a miscellaneous code character. Listing these characters as INVALID_CHARacters will signal the computer to note the occurence of such characters but to ignore them in the formulation of the behavior patterns.

(c) BREAK_CHAR -- This character is used to denote the end of a data set. Whenever the BREAK_CHARaracter is encountered within the data field, the pattern formation is ended and a new pattern is started with the next card. Note that the BREAK_CHARacter must be coded within the bounds specified by the BEGIN_FIELD and BEGIN_LENGTH commands.

(d) INVALID -- This specifies the maximum number of characters which can occur in the data other than the three character types (VALID_CHAR, INVALID_CHAR, and BREAK_CHAR) before the program is terminated. If an illegal character is encountered, an error message will be printed. The control card message which would allow 1000 such errors to be counted would read

INVALID = 1000

Output Options:

Once the program is executed and all patterns have been identified, counted, and stored, the patterns are printed according to the following user options:

(a) MOST_FREQUENT -- The user must specify the number of patterns to be listed in the printout. For example, the command

MOST_FREQUENT = 100

specifies that the 100 most frequently occurring patterns be listed. Although this number is very arbitrary, it has been observed that, beyond the top 50 patterns, the frequency of individual patterns drops off rapidly.

(b) ALPHA_ORDER -- Behavior patterns can be listed alphabetically by the first code (should that code be an alpha character) by using the control card message

ALPHA_ORDER = 'YES'

should the user not want an alphabetical listing, the message

ALPHA_ORDER = 'NO'

must be used.

As a final note, the program listed herein is efficient and economical for small amounts of data (less than 22,000 raw behavior codes). Beyond that point, the program becomes costly to run

because all the data are stored in core. For larger amounts of raw data, an alternate program is available which will handle up to 80,000 raw behavior codes. Persons interested in the larger capability should contact the authors for further information.

REFERENCES

Campbell, J.R. Pattern Analysis - A macroscopic development for interaction analysis. Paper delivered at the annual meeting of the National Association for Research in Science Teaching, Detroit, March 27-29, 1973.

Flanders, N.A. Teacher influence, pupil attitudes, and achievement. Cooperative Research Monograph No. 12, Department of Health, Education, and Welfare, 1965, O.E. - 25040.

Penick, J.E.; Shymansky, J.A.; Matthews, C.C.; and Good, R.G. Studying the effects of two quantitatively defined teaching strategies on student behavior in elementary school science using macroanalytic techniques. Journal of Research in Science Teaching, 1976, 13 (4), 289-296.

Rosenshine, B. Experimental studies of indirect teaching. Classroom Interaction Newsletter, 1970, 5 (2), 7-10.

Shymansky, J.A.; Penick, J.E.; Good, R.G.; and Matthews, C.C. Using macroanalytic techniques to study teacher behavior patterns. Journal of Research in Science Teaching, 1975, 12 (3), 221-227.

Smith, B.O. Teaching: Conditions of its evaluation. in The Evaluation of Teaching. A report to the Second P. Lambda Theta Catena, Washington, D.C., 1967, 65-84.

Withall, J. The development of a technique for the measurement of social-emotional climate in classrooms. Journal of Experimental Education, 1949, 17, 347-361.

Withall, J. The development climate index. Journal of Educational Research, 1951, 45 (2), 93-100.

APPENDIX I

SUMMARY OF COMPUTER KEY WORDS AND USER OPTIONS

COMPUTER KEYWORD/ OPTION LIST	DEFAULT VALUE	RESTRICTIONS
BEGIN_FIELD	1	1 ≤ BEGIN_FIELD ≤ 80
BEGIN_LENGTH	80	1 ≤ BEGIN_LENGTH ≤ 80
FIELD_SPAN	1	1 ≤ FIELD_SPAN ≤ 80
VALID_CHAR	null string	1 ≤ VALID_CHAR ≤ 40
INVALID_CHAR	null string	INVALID_CHAR ≤ 10
BREAK_CHAR	'/'	Only one BREAK_CHAR allowed
PATLEN	2	2 ≤ PATLEN ≤ 5
COLLAPSE	'NO'	Checks to see if it is YES -- otherwise it is NO
INVALID	90	INVALID ≤ 32,767
MOST_FREQUENT	20	1 ≤ MOST_FREQUENT ≤ 100
ALPHA_ORDER	'NO'	Checks to see if it is YES -- otherwise it is NO

APPENDIX II

SAMPLE PRINTOUT

```
                                                    H A S P    S Y S T E M    L O G

 13.32.44 JOB 184 -- MACRO    -- BEGINNING EXEC - INIT 8 - CLASS D
 13.34.06 JOB 184 END EXECUTION.


//MACRO  JOB (--------,5,,1101),'SHYMANSKY'                                  JOB 184
//  EXEC  PL1CLG,PARM.PL1L='SM=(2,80,1)',
//  REGION.GO=200K
//PL1L.SYSIN DD *
IEF142I - STEP WAS EXECUTED - COND CODE 0004

ACCTNG --            17.05 SEC. CPU,          17.25 SEC. WAIT,  DAC=    85,  HWM= 100K

IEF142I - STEP WAS EXECUTED - COND CODE 0004

ACCTNG --             2.85 SEC. CPU,          29.40 SEC. WAIT,  DAC=   468,  HWM=  98K

//GO.SEQOUT DD SYSOUT=A
//GO.ORDOUT DD SYSOUT=A
//GO.SYSIN DD *
//GO.CARD DD *
IEF142I - STEP WAS EXECUTED - COND CODE 0000

ACCTNG --             5.55 SEC. CPU,           8.17 SEC. WAIT,  DAC=     0,  HWM= 110K
```

13

'ERSION 5 5 OS/360 PL/I COMPILER (F)

'L/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

SM=(2,80,1)

'HE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

```
  EBCDIC
  CHAR60
NOMACRO
  SOURCE2
NOMACDCK
  COMP
  SOURCE
NOATR
NOXREF
NOEXTREF
NOLIST
  LOAD
NODECK
  FLAGW
  STMT
  SIZE=0096336
  LINECNT=060
  OPT=01
  SORMGIN=(002,080,001)
NOEXTDIC
  NEST
  OPLIST
  SYNCHKT
```

14

O P T I O N S I N E F F E C T

EBCDIC, CHAR60, NOMACRO, SOURCE2, NOMACDCK, COMP, SOURCE, NOATR, NOXREF,

NOEXTREF, NOLIST, LOAD, NODECK, FLAGW, STMT, SIZE=0096336, LINECNT=060, OPT=01,

SORMGIN=(002,080,001), NOEXTDIC, NEST, OPLIST, SYNCHKT

```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST
  1                    PRNT PROC OPTIONS(MAIN)
  2     1                  DCL 1 NODE BASED (NLINKPT),
                               2 LLINK POINTER,
                               2 RLINK POINTER,
                               2 BALLANCE FIXED BIN(15),
                               2 KEY   CHAR(5),
                               2 COUNT FIXED BIN(15),
                               2 LENGT(5) FIXED BIN(15)
  3     1                  DCL 1 HEAD    BASED (HEADPT),
                               2 NLLPT POINTER,
                               2 ROOTPT POINTER,
                               2 HIGHT FIXED BIN(15)
  4     1                  DCL A          FIXED BIN(15)
  5     1                  DCL ALL CHARS CHAR(51) VARYING INIT('')
  6     1                  DCL ALPHA ORDER CHAR(3) INIT('NO')
  7     1                  DCL BEGIN FIELD FIXED BIN(15) INIT(1)
  8     1                  DCL BEGIN LENGTH FIXED BIN(15) INIT(80)
  9     1                  DCL BREAK CHAR CHAR(1) VARYING INIT('/')
 10     1                  DCL BREAKBEGIN FIXED BIN(15)
 11     1                  DCL BREAKCNT  FIXED BIN(15)
 12     1                  DCL CARD      FILE
 13     1                  DCL CARDS     FIXED BIN(15) INIT(0)
 14     1                  DCL CNT       FIXED BIN(15)
 15     1                  DCL COLLAP    FIXED BIN(15) INIT(0)
 16     1                  DCL COLLAPSE  CHAR(3) INIT('NO ')
 17     1                  DCL DROT      FIXED BIN(15) INIT(0)
 18     1                  DCL EOF       FIXED BIN(15) INIT(0)
```

```
19    1          DCL FIELD SPAN FIXED BIN(15) INIT(1)
20    1          DCL FIRST(0 51) FIXED BIN(31) INIT((52)0)
21    1          DCL FIELDEND  FIXED BIN(15)
22    1          DCL FIRST NODE FIXED BIN(15) INIT(1)
23    1          DCL FIRST TIME FIXED BIN(15) INIT(1)
24    1          DCL FIRSTCNT  FIXED BIN(15) INIT(0)
25    1          DCL FREQUENT(50)  POINTER
26    1          DCL HEADPT    POINTER
27    1          DCL INERR     FIXED BIN(15) INIT(0)
28    1          DCL INPT      CHAR(80) INIT((80)' ')
29    1          DCL INVALID   FIXED BIN(31) INIT(50)
30    1          DCL INVALID CHAR CHAR(10) VARYING INIT('')
31    1          DCL INVALIDBEGIN FIXED BIN(15)
32    1          DCL INVALIDCNT  FIXED BIN(15) INIT(0)
33    1          DCL INVLDCNT  FIXED BIN(15)
34    1          DCL LASTCHAR  CHAR(1)
35    1          DCL MOST FREQUENT FIXED BIN(15) INIT(20)
36    1          DCL NCHAR     CHAR(1) INIT(' ')
37    1          DCL NCOMP     FIXED BIN(15) INIT(0)
38    1          DCL NLINKPT   POINTER
39    1          DCL NPOSIT    FIXED BIN(15) INIT(81)
40    1          DCL ORDCNT    FIXED BIN(15) INIT(0)
41    1          DCL ORDLST(101) POINTER
42    1          DCL ORDOUT    FILE PRINT
43    1          DCL P         POINTER
44    1          DCL PAT       CHAR(5) INIT('')
45    1          DCL PATCNT(5) FIXED BIN(15) INIT (0,0,0,0,0)
46    1          DCL PATFND    FIXED BIN(31)
47    1          DCL PATLEN    FIXED BIN(15) INIT(2)
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

 48     1            DCL PERCNT     FLOAT BIN(51)
 49     1            DCL PERCNT1(5) FLOAT BIN(51)
 50     1            DCL PTHIGHT    FIXED BIN(15)
 51     1            DCL PTSTACK(30) POINTER
 52     1            DCL Q          POINTER
 53     1            DCL R          POINTER
 54     1            DCL S          POINTER
 55     1            DCL SEQOUT     FILE PRINT
 56     1            DCL SLSH       FIXED BIN(15)
 57     1            DCL SROT       FIXED BIN(15) INIT(0)
 58     1            DCL START      FIXED BIN(15)
 59     1            DCL T          POINTER
 60     1            DCL TITLES(5) CHAR(6) INIT( ' FIRST','SECOND',' THIRD',
                                        'FOURTH',' FIFTH')
 61     1            DCL TMPORD1    POINTER
 62     1            DCL TMPORD2    POINTER
 63     1            DCL TOT        FIXED BIN(15)
 64     1            DCL TOTAL      FIXED BIN(31) INIT(0)
 65     1            DCL TOTCHAR    FLOAT BIN(51) INIT(0)
 66     1            DCL TOTPER     FLOAT BIN(51)
 67     1            DCL VALID      FIXED BIN(15)
 68     1            DCL VALID CHAR CHAR(40) VARYING INIT('')
 69     1            DCL VALIDBEGIN FIXED BIN(15)  INIT(1)
 70     1            DCL VALIDCNT   FIXED BIN(15)
 71     1            DCL VALIDEND   FIXED BIN(15)
```

```
72    1          ON ERROR PUT DATA

74    1       ON ENDFILE(CARD) BEGIN
76    2                           EOF = 1
77    2                           GO TO CHNG
78    2                        END

79    1       ON ENDPAGE(ORDOUT) BEGIN
81    2          IF COLLAP = 0 THEN
82    2          PUT FILE(ORDOUT) EDIT('NAME','COUNT','FREQUENCY')
                              (SKIP,A(4),X(4),A(5),X(2),A(9))
83    2          ELSE
83    2          PUT FILE(ORDOUT) EDIT('NAME','COUNT','FREQUENCY',
                              (TITLES(I) DO I = 1 TO PATLEN),'TOTAL')
                              (SKIP,A(4),X(4),A(5),X(2),A(9),X(5),
                              (PATLEN)(X(2),A(6)),X(6),A(5))
84    2       END

85    1       ON ENDPAGE(SEQOUT) BEGIN
87    2          IF COLLAP = 0 THEN
88    2          PUT FILE(SEQOUT) EDIT('NAME','COUNT','FREQUENCY')
                              (SKIP,A(4),X(4),A(5),X(2),A(9))
89    2          ELSE
89    2          PUT FILE(SEQOUT) EDIT('NAME','COUNT','FREQUENCY',
                              (TITLES(I) DO I = 1 TO PATLEN),'TOTAL')
                              (SKIP,A(4),X(4),A(5),X(2),A(9),X(5),
                              (PATLEN)(X(2),A(6)),X(6),A(5))
90    2       END
91    1       PUT PAGE
```

19

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

 92     1            PUT EDIT('THE FOLLOWING ARE THE EXECUTION PARAMETERS ')(SKIP,A)
 93     1            PUT SKIP(2)

 94     1            GET DATA(VALID CHAR,INVALID CHAR,BREAK CHAR,COLLAPSE,INVALID,
                         PATLEN,BEGIN FIELD,BEGIN LENGTH,FIELD SPAN,MOST FREQUENT,
                         ALPHA ORDER)
 95     1            PUT DATA(VALID CHAR,INVALID CHAR,BREAK CHAR,COLLAPSE,INVALID,
                         PATLEN,BEGIN FIELD,BEGIN LENGTH,FIELD SPAN,MOST FREQUENT,
                         ALPHA ORDER)
 96     1            PUT SKIP(2)

 97     1            VALIDCNT = LENGTH(VALID CHAR)
 98     1            VALIDEND = VALIDCNT
 99     1            IF VALIDCNT = 0 THEN DO
101     1    1          PUT EDIT('ERROR** NO VALID CHARACTERS SPECIFIED IN INPUT')
                           (SKIP,A)
102     1    1          INERR = INERR + 1
103     1    1        END

104     1            ALL CHARS = VALID CHAR
105     1            BREAKBEGIN = VALIDCNT + 1
106     1            BREAKCNT = LENGTH(BREAK CHAR)
107     1            IF BREAKCNT   0 THEN DO
109     1    1          ALL CHARS = ALL CHARS    BREAK CHAR
110     1    1        END
111     1            ELSE DO
112     1    1          BREAKBEGIN = 0
113     1    1          PUT EDIT('WARNING ** NO BREAK CHARACTER SPECIFIED')(SKIP,A)
114     1    1        END
```

```
115     1           INVLDCNT = LENGTH(INVALID CHAR)
116     1           INVALIDBEGIN = VALIDCNT + BREAKCNT + 1
117     1           IF INVLDCNT   0 THEN DO
119     1   1          ALL CHARS = ALL CHARS    INVALID CHAR
120     1   1       END
121     1           ELSE PUT EDIT('WARNING** NO INVALID CHARACTERS, NOT EVEN A BLANK')
                         (SKIP,A)
122     1           FIRSTCNT = LENGTH(ALL CHARS)
123     1           IF PATLEN ) 2   PATLEN   5 THEN DO
125     1   1          PUT EDIT('ERROR - THIS PROGRAM IS SET FOR MAXIM PATTER LENGTH',
                                ' OF 3 CHARACTERS, THE NUMBER INPUT IS ',PATLEN)
                               (SKIP,A,A,F(3))
126     1   1          INERR = INERR + 1
127     1   1       END

128     1           IF SUBSTR(COLLAPSE,1,1) = 'Y' THEN COLLAP = 1

130     1           IF BEGIN FIELD ) 1   BEGIN FIELD   80 THEN DO
132     1   1          PUT EDIT ('ERROR ** VALUE FOR INPUT VARIABLE   BEGIN FIELD  IS ',
                      'INCORRECT.')(SKIP,A,A)
133     1   1          INERR = INERR + 1
134     1   1       END

135     1           IF BEGIN LENGTH ) 1   BEGIN LENGTH   80 THEN DO
137     1   1          PUT EDIT('ERROR ** THE VALUE FOR  BEGIN LENGTH  IS INCORRECT.')
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

                                        (SKIP,A)
138     1    1             INERR = INERR + 1
139     1    1          END

140     1               FIELDEND = BEGIN FIELD + BEGIN LENGTH - 1
141     1               IF FIELDEND   80 THEN DO
143     1    1             PUT EDIT('ERROR ** THE BEGINNING POSITION ON CARD PLUS THE FIELD',
                          ' LENGTH IS GREATER THAN 80')(SKIP,A,A)
144     1    1             INERR = INERR + 1
145     1    1          END

146     1               IF FIELD SPAN ) 1   FIELD SPAN   80 THEN DO
148     1    1             PUT EDIT('THE VALUE FOR  FIELD SPAN IS INCORRECT.')(SKIP,A)
149     1    1             INERR = INERR + 1
150     1    1          END

151     1               IF MOST FREQUENT ) 1   MOST FREQUENT   100 THEN DO
153     1    1             PUT EDIT('WARNING ** THE VALUE FOR VARIABLE  MOST FREQUENT  IS GREATER',
                           ' THAN 100, IT IS RESET TO 100')(SKIP,A,A)
154     1    1             MOST FREQUENT = 100
155     1    1          END
156     1               IF INERR   0 THEN DO
158     1    1             PUT EDIT ('ERROR ** EXECUTION IS TERMINATED BECAUSE OF  ',
                          'PARAMETER ERRORS ')(SKIP,A,A)
159     1    1             GO TO FINI
160     1    1          END
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

161     1          BRK DO WHILE (EOF = 0)
162     1    1         IF FIRST TIME = 1 THEN DO
164     1    2            FIRST TIME = 0
165     1    2            CALL GETCHAR
166     1    2            DO I = 1 TO PATLEN
167     1    3               CALL GETCHAR
168     1    3               CNT = 1
169     1    3               IF COLLAP = 1 THEN DO
171     1    4                  DO WHILE ( NCHAR = LASTCHAR)
172     1    5                     CNT = CNT + 1
173     1    5                     CALL GETCHAR
174     1    5                  END
175     1    4               END
176     1    3               SUBSTR(PAT,I,1) = LASTCHAR
177     1    3               PATCNT(I) = CNT
178     1    3            END
179     1    2         END
180     1    1         ELSE DO
181     1    2            CALL GETCHAR
182     1    2            CNT = 1
183     1    2            IF COLLAP = 1 THEN DO
185     1    3               DO WHILE (NCHAR = LASTCHAR)
186     1    4                  CNT = CNT + 1
187     1    4                  CALL GETCHAR
188     1    4               END
189     1    3            END
190     1    2    CHNG    SUBSTR(PAT,1,PATLEN-1) = SUBSTR(PAT,2,PATLEN-1)
191     1    2            DO I = 1 TO PATLEN-1
192     1    3               PATCNT(I) = PATCNT(I+1)
193     1    3            END
194     1    2            SUBSTR(PAT,PATLEN,1) = LASTCHAR
195     1    2            PATCNT(PATLEN) = CNT
196     1    2         END
```

```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

                        /* THE FOLLOWING  CODE CREATES AND SEARCHES A BALANCED
                           BINARY TREE. INSERTING NODES WHEN NOT ALREADY IN TREE AND
                           BALANCING TREE AFTER INSERTION, WHEN NECESSARY.
                           THE ALGORITHM WAS ADAPTED FROM    THE ART OF COMPUTER
                           PROGRAMMING, VOLUME 3, SORTING AND SEARCHING    BY DONALD
                           E. KNUTH,   ADDISON-WESLEY1975  PAGES 455 - 457

                        */

197     1    1    A0  DO WHILE (FIRST NODE = 1)
198     1    2          FIRST NODE = 0
199     1    2          ALLOCATE HEAD
200     1    2          ALLOCATE NODE
201     1    2          NLINKPT -  COUNT = 1
202     1    2          NLINKPT -  BALLANCE = 0
203     1    2          NLINKPT -  LLINK = NULL
204     1    2          NLINKPT -  RLINK = NULL
205     1    2          NLINKPT -  KEY = PAT
206     1    2          HEADPT -  HIGHT = 1
207     1    2          HEADPT -  ROOTPT = NLINKPT
208     1    2          DO I = 1 TO PATLEN
209     1    3             NLINKPT -  LENGT(I) = PATCNT(I)
210     1    3          END
211     1    2          PATFND = 1
212     1    2          TOTAL = 1
213     1    2          GO TO A11
214     1    2       END
```

```
215     1     1     A1        /*  INITIALIZE   */
                          T = HEADPT
216     1     1           S = HEADPT -  ROOTPT
217     1     1           P = HEADPT -  ROOTPT

218     1     1     A2     /*  COMPARISON    */
                        NCOMP = NCOMP + 1
219     1     1           IF PAT ) P -  KEY THEN GO TO A3
221     1     1           IF PAT   P -  KEY THEN GO TO A4
223     1     1           DO I = 1 TO PATLEN
224     1     2              P -  LENGT(I) = P -  LENGT(I) + PATCNT(I)
225     1     2           END
226     1     1           P -  COUNT = P -  COUNT + 1
227     1     1              PATFND = PATFND + 1
228     1     1           GO TO A11

229     1     1     A3     /*   MOVE  LEFT  */
                        Q = P -  LLINK
230     1     1           IF Q = NULL THEN DO
232     1     2              ALLOCATE NODE
233     1     2              Q = NLINKPT
234     1     2              P -  LLINK = Q
235     1     2              GO TO A5
236     1     2           END
237     1     1           IF Q -  BALLANCE  = 0 THEN DO
239     1     2              T = P
240     1     2              S = Q
```

```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

241     1    2          END
242     1    1          P = Q
243     1    1          GO TO A2

244     1    1     A4       /*   MOVE RIGHT  */
                        Q = P -  RLINK
245     1    1          IF Q = NULL THEN DO
247     1    2             ALLOCATE NODE
248     1    2             Q = NLINKPT
249     1    2             P -  RLINK = Q
250     1    2             GO TO A5
251     1    2          END
252     1    1          IF Q -  BALLANCE  = 0 THEN DO
254     1    2             T = P
255     1    2             S = Q
256     1    2          END
257     1    1          P = Q
258     1    1          GO TO A2

259     1    1     A5       /*   INSERT  */
                        TOTAL = TOTAL + 1
260     1    1          PATFND = PATFND + 1
261     1    1          Q -  LLINK = NULL
262     1    1          Q -  RLINK = NULL
263     1    1          Q -  BALLANCE = 0
264     1    1          Q -  KEY = PAT
265     1    1          DO I = 1 TO PATLEN
266     1    2             Q -  LENGT(I) = PATCNT(I)
267     1    2          END
268     1    1          Q -  COUNT = 1
```

```
269     1     1    A6       /*   ADJUST BALANCE FACTORS  */
270     1     1         IF PAT ) S -  KEY THEN DO
271     1     2            R = S -  LLINK
272     1     2            P = S -  LLINK
273     1     2         END
274     1     1         ELSE DO
275     1     2            R = S -  RLINK
276     1     2            P = S -  RLINK
277     1     2         END
278     1     1         DO WHILE ( P  = Q)
279     1     2         IF PAT ) P -  KEY THEN DO
281     1     3               P -  BALLANCE = -1
282     1     3               P = P-  LLINK
283     1     3            END
284     1     2            ELSE IF PAT   P -  KEY THEN DO
286     1     3               P -  BALLANCE = 1
287     1     3               P = P -  RLINK
288     1     3            END
289     1     2         END

290     1     1    A7       /*    BALANCE ACT   */
291     1     1         IF PAT ) S -  KEY THEN A = -1
292     1     1         ELSE A = 1
293     1     1         IF S -  BALLANCE = 0 THEN DO
```

```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

295     1    2           S -  BALLANCE = A
296     1    2           HEADPT -  HIGHT = HEADPT -  HIGHT + 1
297     1    2           GO TO A11
298     1    2        END
299     1    1        ELSE DO
300     1    2           IF S -  BALLANCE = -A THEN DO
302     1    3              S -  BALLANCE = 0
303     1    3              GO TO A11
304     1    3           END
305     1    2           ELSE DO
306     1    3              IF S -  BALLANCE = A THEN DO
308     1    4                 IF R -  BALLANCE = A THEN GO TO A8
310     1    4                 ELSE IF R -  BALLANCE = -A THEN GO TO A9
312     1    4              END
313     1    3           END
314     1    2        END
315     1    1        PUT EDIT (' ERROR IN TREE LOOKUP SECTION A7')(SKIP,A)

316     1    1  A8       /*   SINGLE  ROTATION   */
                     SROT = SROT + 1
317     1    1        P = R
318     1    1        IF A = 1 THEN DO
320     1    2           S -  RLINK = R -  LLINK
321     1    2           R -  LLINK = S
322     1    2        END
323     1    1        ELSE DO
324     1    2           S -  LLINK = R -  RLINK
325     1    2           R -  RLINK = S
326     1    2        END
327     1    1        S -  BALLANCE = 0
328     1    1        R -  BALLANCE = 0
329     1    1        GO TO A10
```

28

```
330     1     1     A9        /*   DOUBLE ROTATION   */
                          DROT = DROT + 1
331     1     1           IF A = 1 THEN DO
333     1     2              P = R -  LLINK
334     1     2              R -  LLINK = P -  RLINK
335     1     2              P -  RLINK = R
336     1     2              S -  RLINK = P -  LLINK
337     1     2              P -  LLINK = S
338     1     2           END
339     1     1           ELSE DO
340     1     2              P = R -  RLINK
341     1     2              R -  RLINK = P -  LLINK
342     1     2              P -  LLINK = R
343     1     2              S -  LLINK = P -  RLINK
344     1     2              P -  RLINK = S
345     1     2           END
346     1     1           IF P -  BALLANCE = A THEN DO
348     1     2              S -  BALLANCE = -A
349     1     2              R -  BALLANCE = 0
350     1     2           END
351     1     1           ELSE DO
352     1     2              IF P -  BALLANCE = 0 THEN DO
```

```
PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

354     1    3              S -  BALLANCE = 0
355     1    3              R -  BALLANCE = 0
356     1    3           END
357     1    2           ELSE DO
358     1    3              IF P -  BALLANCE = -A THEN DO
360     1    4                 S -  BALLANCE = 0
361     1    4                 R -  BALLANCE = A
362     1    4              END
363     1    3           END
364     1    2        END
365     1    1        P -  BALLANCE = 0

366     1    1    A10     /*  FINISHING TOUCH  */
367     1    1        IF S = T -  RLINK THEN T -  RLINK = P
368     1    1        ELSE T -  LLINK = P

369     1    1     A11 END BRK
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

370     1           PRNT
                        PUT PAGE
371     1                PUT EDIT ('     THE TOTAL NUMBER OF PATTERNS FOUND IS  ',PATFND)
                              (SKIP,A,F(7))
372     1               PUT EDIT('     THE TOTAL NUMBER OF DIFFERENT PATTERNS FOUND IS  ',TOTAL)
                             (SKIP,A,F(7))
373     1               PUT EDIT(' CHARACTER','COUNT','FREQUENCY')
                                 (SKIP(3),A(10),X(10),A(5),X(3),A)
374     1               DO I = 1 TO VALIDEND
375     1    1             PERCNT = FIRST(I) / TOTCHAR
376     1    1             PUT EDIT(SUBSTR(ALL CHARS,I,1),FIRST(I),PERCNT)
                              (SKIP,X(5),A,X(10),F(7),X(5),F(9,6))
377     1    1           END
378     1               DO I = VALIDEND + 1 TO FIRSTCNT
379     1    1             PUT EDIT (SUBSTR(ALL CHARS,I,1),FIRST(I))
                              (SKIP,X(5),A,X(10),F(7))
380     1    1           END
381     1               PUT EDIT('INVALID',FIRST(0))(SKIP,X(5),A,X(4),F(7))
382     1               IF SUBSTR(ALPHA ORDER,1,1) = 'Y' THEN SIGNAL ENDPAGE(SEQOUT)

                            /* THE FOLLOWING CODE TRAVERSES THE BINARY TREE,
                               PRINTING EACH NODE IN INORDER SEQUENCE.  THE ALGORITHM
                               WAS ADAPTED FROM    THE ART OF COMPUTER PROGRAMMING,
                               VOLUME 1, FUNDAMENTAL ALGORITHMS   BY DONALD E. KNUTH
                               ADDISON-WESLEY, 1975  PAGES 317 - 318
                            */
```

```
384     1          T1      /*  INITIALIZE   */
                       PTHIGHT = 0
385     1              P = HEADPT -  ROOTPT

386     1          T2      /* IS P = NULL              */
387     1              IF P = NULL THEN GO TO T4

388     1          T3      /*  PUT P ON STACK    */
                       PTHIGHT = PTHIGHT + 1
389     1              PTSTACK(PTHIGHT) = P
390     1              P = P -  LLINK
391     1              GO TO T2

392     1          T4      /*POP STACK     */
393     1              IF PTHIGHT - 1 ) 0 THEN GO TO T6
394     1              P = PTSTACK(PTHIGHT)
395     1              PTHIGHT = PTHIGHT - 1

396     1          T5      /*VISIT NODE P    */
397     1              IF ORDCNT = 0 THEN DO
398     1    1            ORDCNT = 1
399     1    1            ORDLST(1) = P
400     1    1         END
401     1              ELSE DO
402     1    1         TMPORD1 = ORDLST(ORDCNT)
403     1    1         IF P -  COUNT  = TMPORD1 -  COUNT THEN DO
405     1    2            IF ORDCNT )  MOST FREQUENT THEN ORDCNT = ORDCNT + 1
407     1    2            DO I = ORDCNT TO 2 BY -1
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

408     1    3              TMPORD1 = ORDLST(I - 1)
409     1    3              IF TMPORD1 -  COUNT  = P -  COUNT THEN DO
411     1    4                 ORDLST(I) = P
412     1    4                 GO TO T5A
413     1    4              END
414     1    3           ORDLST(I) = ORDLST(I - 1)
415     1    3           END
416     1    2           ORDLST(1) = P
417     1    2        END
418     1    1        ELSE DO
419     1    2           IF ORDCNT ) MOST FREQUENT THEN DO
421     1    3              ORDCNT = ORDCNT + 1
422     1    3              ORDLST(ORDCNT) = P
423     1    3           END
424     1    2        END
425     1    1        END
426     1          T5A
427     1             IF SUBSTR(ALPHA ORDER,1,1) = 'Y' THEN DO
428     1    1        TOT = 0
429     1    1        DO K = 1 TO PATLEN
430     1    2           PERCNT1(K) = P -  LENGT(K) / P -  COUNT
431     1    2           TOT = TOT + P -  LENGT(K)
432     1    2        END
433     1    1        TOTPER = TOT / P -  COUNT
434     1    1        PERCNT = P -  COUNT / PATFND
435     1    1        IF COLLAP = 0 THEN PUT FILE(SEQOUT) EDIT(SUBSTR(P -  KEY,1,PATLEN),
                                    P -  COUNT,PERCNT)
                                    (SKIP,A(PATLEN),X(5-PATLEN),F(7),X(3),F(9,7))
```

```
437     1     1          ELSE PUT FILE(SEQOUT) EDIT(SUBSTR(P -  KEY,1,PATLEN),
                                 P    COUNT,PERCNT,(PERCNT1(I) DO I=1 TO PATLEN),
                                 TOTPER)
                                 (SKIP,A(PATLEN),X(5-PATLEN),F(7),X(3),F(9,7),X(5),
                                 (PATLEN)(X(2),F(6,1)),X(4),F(7,1))
438     1     1          END
439     1                P = P -  RLINK
440     1                GO TO T2

441     1           T6       /*FINISH PRINTING  */
                         SIGNAL ENDPAGE(ORDOUT)
442     1                DO I = 1 TO ORDCNT
443     1     1             TMPORD1 = ORDLST(I)
444     1     1          TOT = 0
445     1     1          DO K = 1 TO PATLEN
446     1     2             PERCNT1(K) = TMPORD1    -  COUNT
447     1     2             PERCNT1(K) = TMPORD1    -  LENGT(K) / TMPORD1    -  COUNT
448     1     2             TOT = TOT + TMPORD1    -  LENGT(K)
449     1     2          END
450     1     1          TOTPER = TOT / TMPORD1    -  COUNT
451     1     1          PERCNT = TMPORD1    -  COUNT / PATEND
452     1     1          IF COLLAP = 0 THEN PUT FILE(ORDOUT) EDIT(SUBSTR(TMPORD1 -  KEY,
                                 1,PATLEN),TMPORD1 -  COUNT,PERCNT)
                                 (SKIP,A(PATLEN),X(5-PATLEN),F(7),X(3),F(9,7))
454     1     1          ELSE PUT FILE(ORDOUT) EDIT(SUBSTR(TMPORD1 -  KEY,1,PATLEN),
                                 TMPORD1 -  COUNT,PERCNT,(PERCNT1(J) DO J = 1 TO PATLEN),
                                 TOTPER)
```

```
                    PRNT  PROC  OPTIONS(MAIN)

STMT LEVEL NEST

          (SKIP,A(PATLEN),  X(5-PATLEN),  F(7),  X(3),  F(9,7),  X(5),

          (PATLEN)(X(2),  F(6,1)),  X(4),  F(7,1))

455       1     1         END
```

```
        PRNT PROC OPTIONS(MAIN)

STMT LEVEL NEST

456     1           GETCHAR  PROC
457     2               LASTCHAR = NCHAR
458     2           INVLD
459     2               IF SLSH  = NPOSIT - FIELD SPAN & SLSH ) NPOSIT THEN DO
460     2   1               FIRST(BREAKBEGIN ) = FIRST(BREAKBEGIN ) + 1
461     2   1               NPOSIT = 81
462     2   1               NCHAR = '''
463     2   1               SLSH = 161
464     2   1               FIRST TIME = 1
465     2   1            GO TO BRK
466     2   1           END
467     2               IF NPOSIT   FIELDEND THEN DO
469     2   1               GET FILE(CARD) EDIT (INPT) (COL(1),A(80))
470     2   1               CARDS = CARDS + 1
471     2   1               NPOSIT = BEGIN FIELD
472     2   1               IF FIELD SPAN   1 THEN DO
474     2   2                   SLSH = INDEX(SUBSTR(INPT,BEGIN FIELD,
                                                BEGIN LENGTH),BREAK CHAR)
475     2   2                   IF SLSH   0 THEN SLSH = BEGIN FIELD + SLSH - 1
477     2   2                   ELSE SLSH = 161
478     2   2               END
479     2   1           END
480     2               NCHAR = SUBSTR(INPT,NPOSIT,1)
481     2               NPOSIT = NPOSIT + FIELD SPAN
482     2               VALID = INDEX(ALL CHARS,NCHAR)
483     2               FIRST(VALID) = FIRST(VALID) + 1
484     2               IF VALID = 0 THEN DO
```

```
486     2    1          PUT EDIT('ERROR - ON CARD',CARDS,' AND IN COLUMN ',NPOSIT-FIELD SPAN,
                                 'THERE IS AN INVALID CHARACTER-- ',NCHAR,
                                 ')--  CHARACTER IS SKIPPED')
                                 (SKIP,A,F(7),A,F(7),A,A,A)
487     2    1          INVALIDCNT = INVALIDCNT + 1
488     2    1          IF INVALIDCNT  = INVALID THEN DO
490     2    2             PUT EDIT('NUMBER OF INVALID CHARACTERS EXCEEDS MAXIMUM SPECIFIED',
                                 INVALID,' PROGRAM TERMINATION')(SKIP,A,F(5),A)
491     2    2             GO TO PRNT
492     2    2          END
493     2    1          GO TO INVLD
494     2    1       END
495     2            ELSE DO
496     2    1          IF VALID )= VALIDEND THEN DO
498     2    2             TOTCHAR = TOTCHAR + 1
499     2    2          END
500     2    1          ELSE DO
501     2    2          IF VALID  = INVALIDBEGIN THEN GO TO INVLD
503     2    2          ELSE DO
504     2    3             NPOSIT = 8
505     2    3             NCHAR = '''
506     2    3             FIRST TIME = 1
507     2    3             GO TO BRK
508     2    3          END
509     2    2          END
510     2    1       END
511     2          END GETCHAR
512     1       FINI  END PRNT
```

37

```
 PRNT PROC OPTIONS(MAIN)

STORAGE REQUIREMENTS.
---------------------

THE STORAGE AREA FOR THE PROCEDURE LABELLED PRNT IS 2020 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 72 IS 224 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 75 IS 224 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 80 IS 272 BYTES LONG.
THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 86 IS 272 BYTES LONG.
THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED GETCHAR IS 352 BYTES LONG.
THE PROGRAM CSECT IS NAMED PRNT      AND IS 14146 BYTES LONG.
THE STATIC  CSECT IS NAMED ***PRNTA AND IS 4480 BYTES LONG.

*STATISTICS*      SOURCE RECORDS =       542,PROG TEXT STMNTS =       512,OBJECT BYTES =     14146
```

38

PRNT PROC OPTIONS(MAIN)

COMPILER DIAGNOSTICS.

WARNINGS

IEM0227I NO FILE/STRING OPTION SPECIFIED IN ONE OR MORE GET/PUT STATEMENTS. SYSIN/SYSPRINT HAS BEEN ASSUMED IN EACH CASE.

IEM0764I ONE OR MORE FIXED BINARY ITEMS OF PRECISION 15 OR LESS HAVE BEEN GIVEN HALFWORD STORAGE. THEY ARE FLAGGED '*********' IN THE XREF/ATR LIST.

IEM3898I COMPILER CORE REQUIREMENT EXCEEDED SIZE GIVEN. AUXILIARY STORAGE USED.

END OF DIAGNOSTICS.

AUXILIARY STORAGE WILL NOT BE USED FOR DICTIONARY WHEN SIZE = 69K

COMPILE TIME .28 MINS

ELAPSED TIME .64 MINS

THE FOLLOWING ARE THE EXECUTION PARAMETERS

VALID CHAR='ELMOQRSTWZ' INVALID CHAR=' ' BREAK CHAR='/' COLLAPSE='YES'

INVALID= 50 PATLEN= 5 BEGIN FIELD= 21

BEGIN LENGTH= 60 FIELD SPAN= 3 MOST FREQUENT= 50

ALPHA ORDER='NO'

THE TOTAL NUMBER OF PATTERNS FOUND IS 1994
THE TOTAL NUMBER OF DIFFERENT PATTERNS FOUND IS 1243

CHARACTER	COUNT	FREQUENCY
E	1796	0.327977
L	885	0.161614
M	221	0.040358
O	295	0.053871
Q	75	0.013696
R	1051	0.191928
S	0	0.000000
T	304	0.055515
W	726	0.132579
Z	123	0.022462
/	29	
	41	
INVALID	0	

NAME	COUNT	FREQUENCY	FIRST	SECOND	THIRD	FOURTH	FIFTH	TOTAL
ELELE	30	0.0150299	3.1	1.5	2.9	1.2	5.9	14.7
LELEL	23	0.0115204	1.3	2.8	1.2	6.2	1.6	13.0
WEWEW	21	0.0105286	3.5	2.0	4.0	2.8	3.5	15.8
EMEME	20	0.0100250	5.3	1.7	4.8	1.7	8.2	21.8
EWEWE	20	0.0100250	2.5	4.1	2.8	4.3	2.9	16.7
_RLRL	19	0.0095215	2.2	2.8	1.7	2.8	1.9	11.5
_OLOL	15	0.0075073	1.6	1.8	1.7	1.4	1.5	8.0
MEMEM	15	0.0075073	2.0	4.3	1.6	4.9	2.1	14.9
RLRLR	15	0.0075073	3.1	1.4	2.9	1.9	2.3	11.7
OLOLO	14	0.0070190	1.6	1.7	1.5	1.5	1.5	7.8
TELEL	11	0.0055084	1.2	5.3	1.4	1.7	1.0	10.5
ELTEL	8	0.0039978	11.3	1.1	1.4	4.4	1.3	19.4
_ETEL	8	0.0039978	1.1	1.8	1.1	2.9	1.9	8.8
_RLRO	8	0.0039978	2.1	2.1	1.1	1.5	1.1	8.0
_TELE	8	0.0039978	1.1	1.3	4.1	1.3	4.4	12.1
_TLTL	8	0.0039978	2.8	1.3	1.9	1.1	1.4	8.4
_WLWL	8	0.0039978	1.3	2.8	1.0	2.4	2.6	10.0
ELETE	7	0.0035095	4.6	1.3	3.4	1.1	5.1	15.6
ERERE	7	0.0035095	5.3	2.1	3.0	7.6	3.4	21.4
ETELE	7	0.0035095	3.3	1.1	5.3	1.7	2.4	13.9
LELEO	7	0.0035095	1.6	6.6	1.3	2.7	1.4	13.6
_ELTE	7	0.0035095	1.4	11.0	1.3	1.4	3.9	19.0
OLOLR	7	0.0035095	1.6	1.1	1.4	2.0	3.1	9.3
RERER	7	0.0035095	1.7	2.9	2.1	2.7	6.6	16.0
ELTLE	6	0.0030060	2.3	1.0	1.5	1.5	3.3	9.7

42

LELET	6	0.0030060	1.7	3.3	1.0	1.8	1.0	8.8
LRLOL	6	0.0030060	2.2	2.3	4.7	1.7	2.0	12.8
ROLOL	6	0.0030060	2.8	1.0	1.3	1.5	1.5	8.2
RWLRL	6	0.0030060	3.8	5.2	1.3	2.3	1.3	14.0
TLTLT	6	0.0030060	1.8	1.7	1.5	1.3	2.0	8.3
WLWLR	6	0.0030060	2.7	1.0	1.8	2.8	2.5	10.8
ELELO	5	0.0025024	2.2	1.0	1.4	1.6	1.2	7.4
ELELT	5	0.0025024	5.2	1.6	15.6	1.4	1.4	25.2
ELEOE	5	0.0025024	10.6	1.6	11.2	1.0	2.2	26.6
ELEOL	5	0.0025024	2.0	1.2	5.2	1.6	1.6	11.6
ELETL	5	0.0025024	5.4	1.0	1.2	1.2	1.4	10.2
EOLOL	5	0.0025024	2.4	1.6	1.4	2.0	1.4	8.8
LELTL	5	0.0025024	1.4	2.2	1.4	1.2	1.0	7.2
LOLRL	5	0.0025024	1.8	1.2	2.8	2.4	1.8	10.0
LRORO	5	0.0025024	1.0	4.0	1.0	4.2	1.0	11.2
OLRLR	5	0.0025024	1.4	3.2	2.4	1.2	3.8	12.0
RLRLW	5	0.0025024	4.6	1.0	3.8	1.4	3.6	14.4
RLROL	5	0.0025024	1.4	1.4	1.4	1.4	1.4	7.0
RORDL	5	0.0025024	4.0	1.4	3.4	1.0	1.8	11.6
RWRWR	5	0.0025024	4.6	3.0	3.0	3.2	2.2	16.0
TELTE	5	0.0025024	1.2	1.6	1.2	1.0	2.4	7.4
TLETE	5	0.0025024	1.8	1.0	1.2	1.2	3.2	8.4
TLTLE	5	0.0025024	1.2	1.4	1.8	1.0	1.4	6.8
WLRLR	5	0.0025024	1.8	2.8	2.2	2.0	1.8	10.6
WLWLW	5	0.0025024	2.4	1.0	2.4	1.4	2.4	9.6

* * * JOB ACCOUNTING SUMMARY * * *

JOB NAME MACRO PROGRAMMER ID SHYMANSKY

ITEM	QUANTITY, THIS RUN	QUANTITY, ACCUMULATED
COMPUTE TIME	25.45 SEC.	4 MIN. 47.44 SEC.
WAIT TIME	54.82 SEC.	12 MIN. 58.86 SEC.
CORE STORAGE	9.3 MEGABYTE-SECONDS	118.8 MEGABYTE-SECONDS
DIRECT ACCESS USAGE	553 I/O ACCESSES	6,806 I/O ACCESSES
CARDS IN	835 CARDS	7,746 CARDS
PAGES OUT	22 PAGES	354 PAGES
LINES OUT	750 LINES	11,562 LINES
PRINT COSTS, THIS LISTING	$0.29	
TOTAL RUN COSTS	$3.74	